

UNIT27: SKINNED MESH – CONTROLLING ANIMATION

【学習要項】

□Controlling animation

【演習手順】

1. framework クラスの initialize メンバ関数で skinned_mesh コンストラクタ引数を.¥¥resources¥¥plantune.fbx に変更する
※plantune.fbx は右手系・Y 軸アップ・センチメートル単位・三角形化済み

2. animation::keyframe::node 構造体にメンバ変数を追加する

```
1: struct animation
2: {
3:     std::string name;
4:     float sampling_rate{ 0 };
5:
6:     struct keyframe
7:     {
8:         struct node
9:         {
10:            // 'global_transform' is used to convert from local space of node to global space of scene.
11:            DirectX::XMFLOAT4X4 global_transform{ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 };
12:
13:            // The transformation data of a node includes its translation, rotation and scaling vectors
14:            // with respect to its parent.
15:            DirectX::XMFLOAT3 scaling{ 1, 1, 1 };
16:            DirectX::XMFLOAT4 rotation{ 0, 0, 0, 1 }; // Rotation quaternion
17:            DirectX::XMFLOAT3 translation{ 0, 0, 0 };
18:        };
19:        std::vector<node> nodes;
20:    };
21:    std::vector<keyframe> sequence;
22:};
```

3. skinned_mesh クラスの fetch_animations メンバ関数を変更する

```
1: void skinned_mesh::fetch_animations(FbxScene* fbx_scene,
2:     vector<animation>& animation_clips, float sampling_rate)
3: {
4:     FbxArray<FbxString*> animation_stack_names;
5:     fbx_scene->FillAnimStackNameArray(animation_stack_names);
6:     const int animation_stack_count{ animation_stack_names.GetCount() };
7:     for (int animation_stack_index = 0; animation_stack_index < animation_stack_count;
8:         ++animation_stack_index)
9:     {
10:         :
11:         :
12:         :
13:         for (FbxTime time = start_time; time < stop_time; time += sampling_interval)
14:         {
15:             animation::keyframe& keyframe{ animation_clip.sequence.emplace_back() };
16:
17:             size_t node_count{ scene_view.nodes.size() };
18:             keyframe.nodes.resize(node_count);
19:             for (size_t node_index = 0; node_index < node_count; ++node_index)
20:             {
21:                 FbxNode* fbx_node
22:                 { fbx_scene->FindNodeByName(scene_view.nodes.at(node_index).name.c_str()) };
23:                 if (fbx_node)
24:                 {
25:                     animation::keyframe::node& node{ keyframe.nodes.at(node_index) };
26:                     // 'global_transform' is a transformation matrix of a node with respect to
27:                     // the scene's global coordinate system.
28:                     node.global_transform = to_xmfloat4x4(fbx_node->EvaluateGlobalTransform(time));
29:
30:                     // 'local_transform' is a transformation matrix of a node with respect to
31:                     // its parent's local coordinate system.
32:                     const FbxAMatrix& local_transform{ fbx_node->EvaluateLocalTransform(time) };
33:                     node.scaling = to_xmfloat3(local_transform.GetS());
```

UNIT27: SKINNED MESH – CONTROLLING ANIMATION

```
*34:             node.rotation = to_xmfloat4(local_transform.GetQ());
*35:             node.translation = to_xmfloat3(local_transform.GetT());
36:         }
37:     }
38: }
39: }
40: for (int animation_stack_index = 0; animation_stack_index < animation_stack_count;
41:     ++animation_stack_index)
42: {
43:     delete animation_stack_names[animation_stack_index];
44: }
45: }
```

4. skinned_mesh クラスに update_animation メンバ関数を追加する

```
1: void skinned_mesh::update_animation(animation::keyframe& keyframe)
2: {
3:     size_t node_count{ keyframe.nodes.size() };
4:     for (size_t node_index = 0; node_index < node_count; ++node_index)
5:     {
6:         animation::keyframe::node& node{ keyframe.nodes.at(node_index) };
7:         XMATRIX S{ XMMatrixScaling(node.scaling.x, node.scaling.y, node.scaling.z) };
8:         XMATRIX R{ XMMatrixRotationQuaternion(XMLoadFloat4(&node.rotation)) };
9:         XMATRIX T{ XMMatrixTranslation(node.translation.x, node.translation.y, node.translation.z) };
10:
11:         int64_t parent_index{ scene_view.nodes.at(node_index).parent_index };
12:         XMATRIX P{ parent_index < 0 ? XMMatrixIdentity() :
13:             XMLoadFloat4x4(&keyframe.nodes.at(parent_index).global_transform) };
14:
15:         XMStoreFloat4x4(&node.global_transform, S * R * T * P);
16:     }
17: }
```

5. framework クラスの render メンバ関数を変更する

※動作確認後#if-#endif ディレクティブのコードは無効にすること(17:-22:行目)

```
1: int clip_index{ 0 };
2: int frame_index{ 0 };
3: static float animation_tick{ 0 };
4:
5: animation& animation{ skinned_meshes[0]->animation_clips.at(clip_index) };
6: frame_index = static_cast<int>(animation_tick * animation.sampling_rate);
7: if (frame_index > animation.sequence.size() - 1)
8: {
9:     frame_index = 0;
10:    animation_tick = 0;
11: }
12: else
13: {
14:    animation_tick += elapsed_time;
15: }
16: animation::keyframe& keyframe{ animation.sequence.at(frame_index) };
*17: #if 1
*18:    XMStoreFloat4(&keyframe.nodes.at(30).rotation,
*19:        DirectX::XMQuaternionRotationAxis(DirectX::XMVectorSet(1, 0, 0, 0), 1.5f));
*20:    keyframe.nodes.at(30).translation.x = 300.0f;
*21:    skinned_meshes[0]->update_animation(keyframe);
*22: #endif
23:    skinned_meshes[0]->render(immediate_context.Get(), world, material_color, &keyframe);
```

6. 実行し、プランチェーン待機モーションをとりながら、首が伸び、さらに左を向いていることを確認する

7. 5. 19: 20:行目の定数値を変数に変更し、ImGui から実行時に変更できるようにする

【評価項目】

☐ アニメーション制御